Accelerating Dense Linear Algebra on the GPU

Sørensen, Hans Henrik Brandenborg

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Accelerating Dense Linear Algebra on the GPU

Hans Henrik Brandenborg Sørensen, Postdoc

Section for Scientific Computing

DTU Informatics

DTU Informatics

Department of Informatics and Mathematical Modeling

GPUlab DTU Informatics

GPU-LABoratory

Research and education in Graphics Processing Units in Denmark

Established in August 2008 and is a unique national competence center and hardware laboratory.

- Development of efficient algorithms
- High-performance scientific computing
- Performance profiling and prediction
- Software development
- Education

http://gpulab.imm.dtu.dk

With support 2010-2013 from national FTP grant
"*Desktop Computing on Consumer Graphics Cards*"
PI: Prof. Per Christian Hansen

GPUlab
DTU Informatics

Facebook

Prof.
Per C. Hansen

Assoc. Prof.
Jeppe R. Frisvad

Assoc. Prof.
Bernd Dammann

Assoc. Prof.
John B. Jørgensen

Postdoc
Hans-Henrik B. Sørensen

PhD Student
Stefan L. Glimberg

PhD Student
Nicolai Fog Gade-Nielsen

Assoc. Prof.
Allan P. Engsig-Karup

GPUlab
DTU Informatics

Educational activities



Topics:
- Parallel programming
- High-Performance Computing
- Algorithms and Numerical Methods for Solving Differential Equations

~200 participants with approx. 150 with research background since May 2010

2010

- Ph.D. School on "Scientific GPU Computing" with Hendrik Lensch (Germany), Robert Strzodka (Germany) and Timothy Lanfear (Nvidia)

2011

- One week course on "Introduction to parallel programming of GPUs using CUDA" in DTU course 02614 on "High-Performance Computing"
- Ph.D. School in "Scientific GPU Computing" with Tim Warburton (Rice University, USA)
- Ph.D. School in "Iterative Methods for Large Linear Systems" with Tim Kelley (North Carolina University)
- Workshop on "Easy, Effective, Efficient: GPU Programming in Python with PyOpenCL and PyCUDA" with Andreas Kloeckner (New York University, USA)

Outline

1. Motivation
2. Identifying tuning parameters / kernel designs
3. Performance prediction
4. Auto-tuning results

Motivation

Motivation

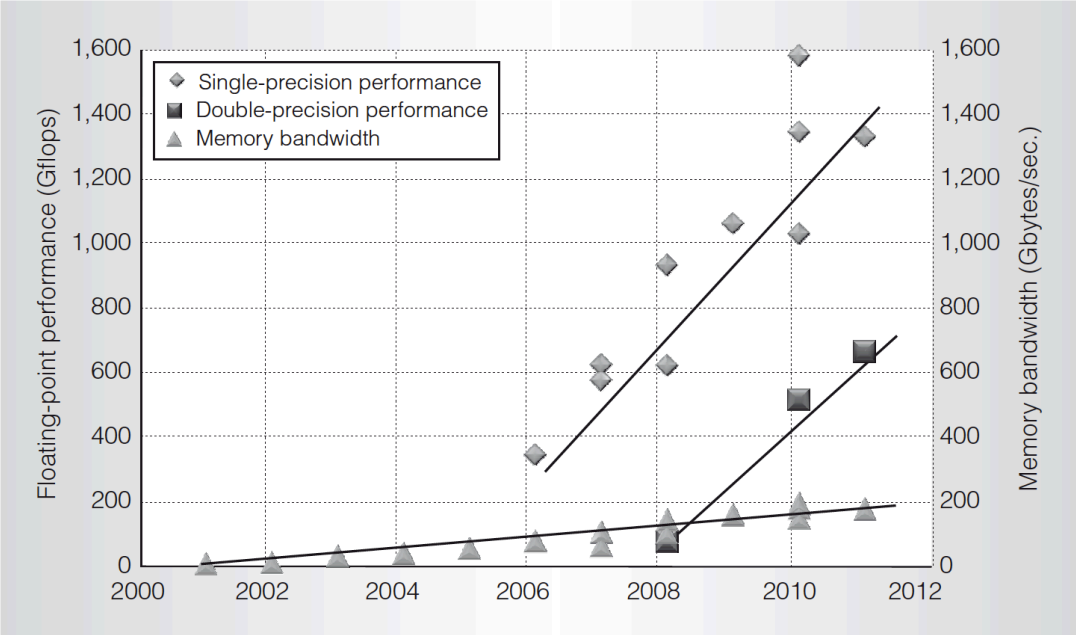
- Running time of dense linear algebra depends on;
  - # flops * time per flop
  - # bytes moved / bandwidth
  - # messages * latency

Motivation

- Running time of dense linear algebra depends on;
  - # flops * time per flop — Compute bound
  - # bytes moved / bandwidth
  - # messages * latency — Memory bound

Motivation

- Running time of dense linear algebra depends on;
  - # flops * time per flop — Compute bound
  - # bytes moved / bandwidth
  - # messages * latency — Memory bound



Source: Keckler et al, "GPUs and the future of parallel computing", Nvidia

Motivation


WANTED
• High memory-bandwidth
(good for memory-bound algorithms)
• Massive multi-threading capability
(good latency-hiding)
• Many-core processing
(good for massively parallel processing)
• Programmable with standard tools
(good for productivity)
• Mass-produced commodity hardware
(widely available and affordable)

High-Performance Computing on GPUs

PRESENTED BY UNIVERSITY OF MANNHEIM

ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

BERKELEY LAB Lawrence Berkeley National Laboratory

FIND OUT MORE AT www.top500.org

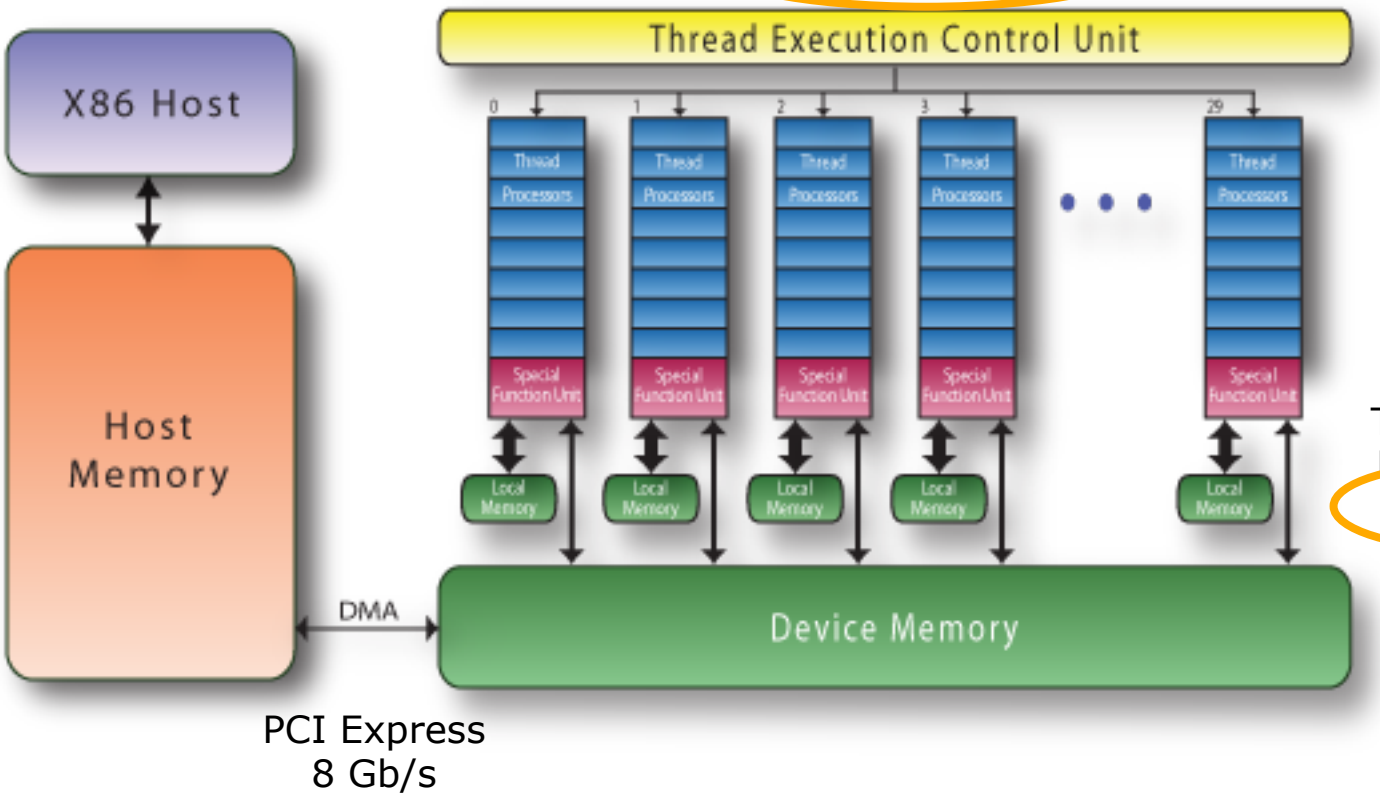
	NAME/MANUFACTURER/COMPUTER	SITE	COUNTRY	CORES	R_{max} Pflop/s
1	**K computer** SPARC64 VIIIfx 2.0GHz, Tofu interconnect	RIKEN	Japan	705,024	10.5
2	**Tianhe-1A** 6-core Intel X5670 2.93 GHz + Nvidia M2050 GPU w/custom interconnect	NUDT/NSCC/Tianjin	China	186,368	2.57
3	**Jaguar** Cray XT-5 6-core AMD 2.6 GHz w/custom interconnect	DOE/OS/ORNL	USA	224,162	1.76
4	**Nebulae** Dawning TC3600 Blade Intel X5650 2.67 GHz + NVidia Tesla C2050 GPU w/ Iband	NSCS	China	120,640	1.27
5	**Tsubame 2.0** HP Proliant SL390s G7 nodes (Xeon X5670 2.93GHz) , NVIDIA Tesla M2050 GPU w/Iband	TiTech	Japan	73,278	1.19

GPUlab DTU Informatics

GPULab hardware

Host
Intel Xeon
4 cores
2.4 GHz
38 Gflop/s (DP)

Accelerator (GPU)
Nvidia C2050 "Fermi"
448 cores
~~1.15 GHz~~
515 Gflop/s (DP)



Theoretical ~~bandwidth~~
144 Gb/s

Source: PGI Insider: "The PGI Accelerator Programming Model on NVIDIA GPUs. Part 1"

Dense Linear Algebra on GPUs

- BLAS (Basic Linear Algebra Subroutines)
  - Level 1 BLAS: (xAXPY, xDOT, xNRM2, etc.)
    - Vectors of length N (4xN bytes)
    - 2x4xN bytes : O(N) flops
    - Memory bound

Dense Linear Algebra on GPUs

- BLAS (Basic Linear Algebra Subroutines)
  - Level 1 BLAS: (xAXPY, xDOT, xNRM2, etc.)
    - Vectors of length N (4xN bytes)
    - 2x4xN bytes : O(N) flops
    - Memory bound
  - Level 2 BLAS: (xGEMV, xSYMV, xTRSV, etc.)
    - Matrix of size NxN (4xNxN bytes) + Vectors
    - 4xN^2 + 2x4xN bytes : O(N^2) flops
    - Memory bound

Dense Linear Algebra on GPUs

- BLAS (Basic Linear Algebra Subroutines)
  - Level 1 BLAS: (xAXPY, xDOT, xNRM2, etc.)
    - Vectors of length N (4xN bytes)
    - 2x4xN bytes : O(N) flops
    - Memory bound
  - Level 2 BLAS: (xGEMV, xSYMV, xTRSV, etc.)
    - Matrix of size NxN (4xNxN bytes) + Vectors
    - 4xN^2 + 2x4xN bytes : O(N^2) flops
    - Memory bound
  - Level 3 BLAS: (xGEMM, xSYMM, xTRSM, etc.)
    - 1 or 2 matrices of size NxN (4xNxN bytes)
    - (2x)4xN^2 bytes : O(N^3) flops
    - Compute bound for large N

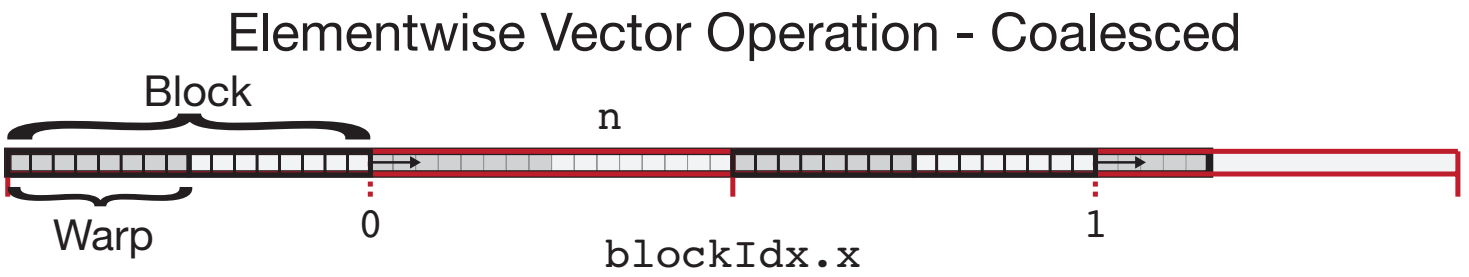
Dense Linear Algebra on GPUs

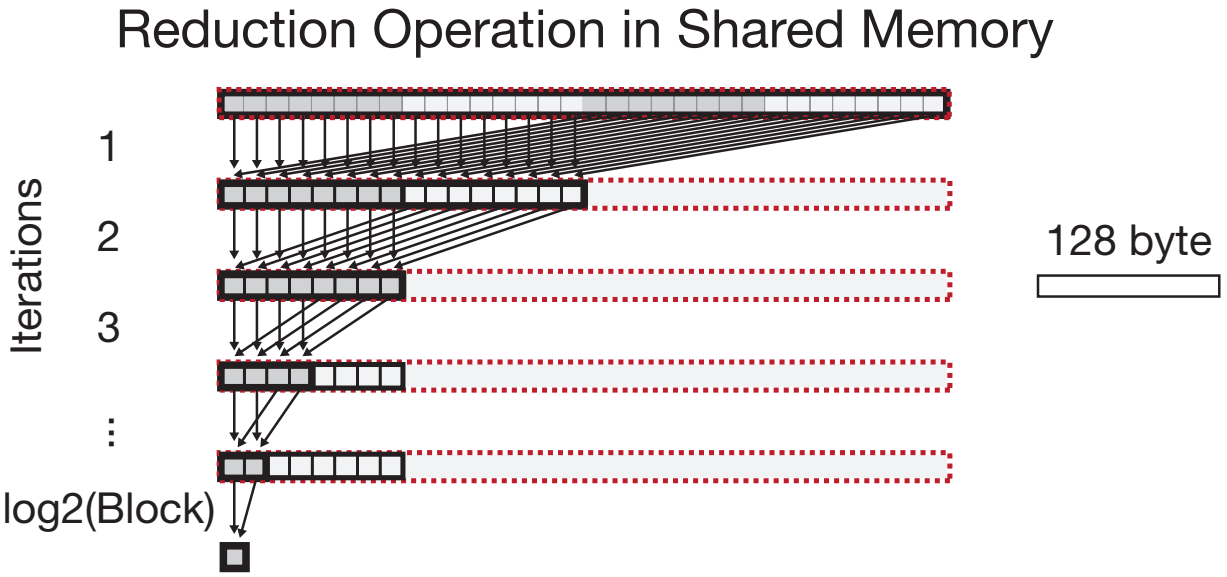
Tesla C2050 (Fermi): 448 CUDA cores @ 1.15GHz; theoretical SP peak=1.03 TFlop/s

- Level 1 BLAS: (xAXPY, xDOT, xNRM2, etc.)
  - Vectors of length N (4xN bytes)
  - 2x4xN bytes : O(N) flops
  - Me
  - ➢ Performance < 144 GB/s ~ 72 GFlops/s
  - ➢ Little attention from the GPU community!
- Level
  - Matrix of size NxN (4xNxN bytes) + Vectors
  - 4xN^2 + 2x4xN bytes : O(N^2) flops
  - Memory bound
- Level 3 BLAS: (xGEMM, xSYMM, xTRSM, etc.)
  - 1 c
  - (2x
  - Compute bound for large N
  - ➢ Performance > 600 GFlops/s.
  - ➢ Very much attention!

Identifying tuning parameters / kernel designs

Level 1 BLAS: Vector Operations





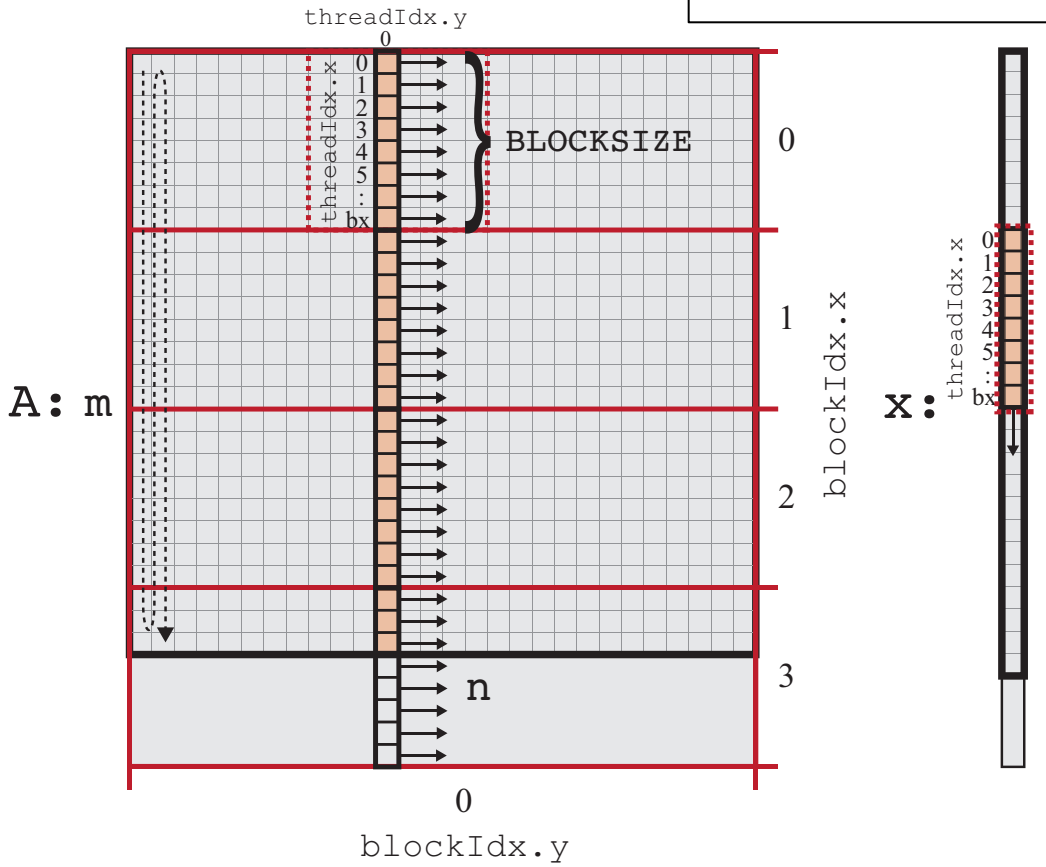
Tuning parameters: `BLOCKSIZE, WORKSIZE_n, UNROLL_LEVEL`

Level 2 BLAS: Matrix-Vector Mult.

MatVec: Typical Ax=y algorithm

- Compute one dot product per thread
- Tuning parameter: `BLOCKSIZE`

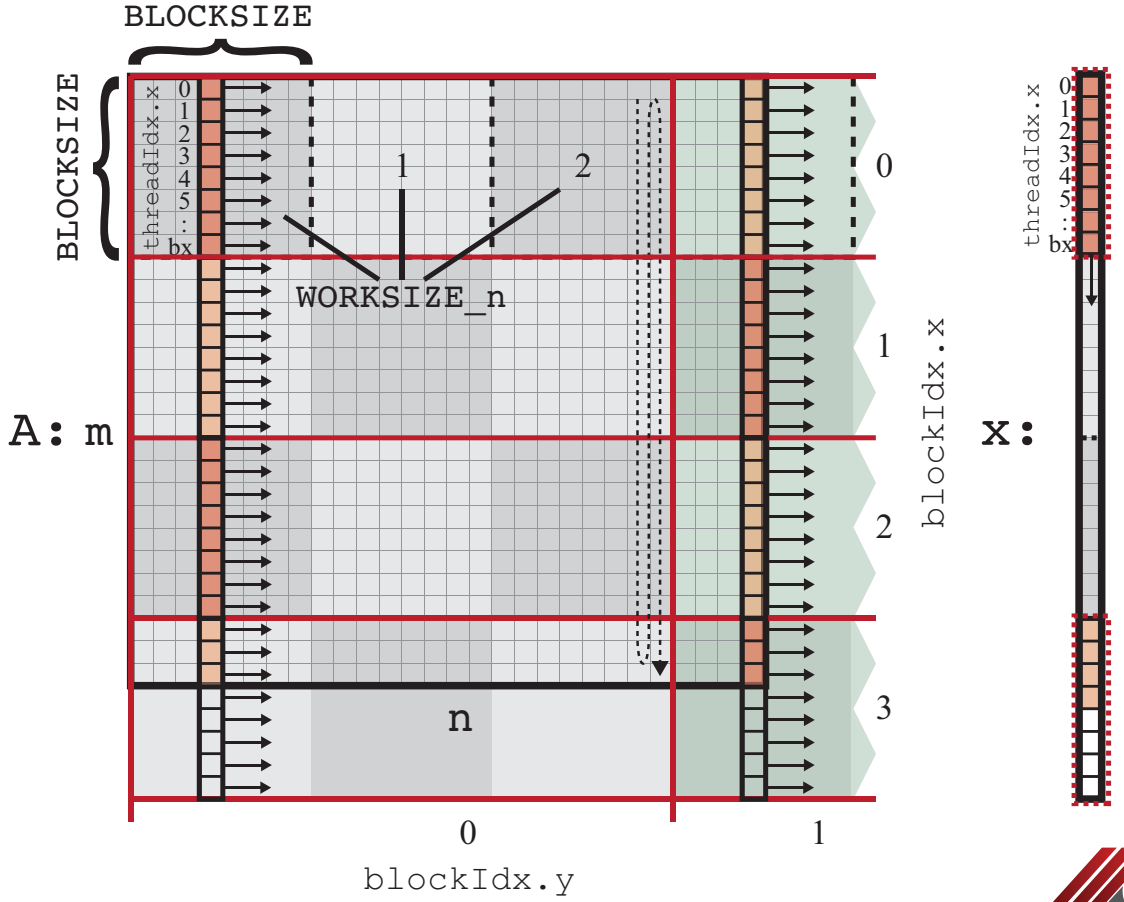
```
for i = 1:Rows
   for j = 1:Columns
      y(i) += A(i,j) * x(j);
   end
end
```



Level 2 BLAS: Matrix-Vector Mult.

Several threads per row

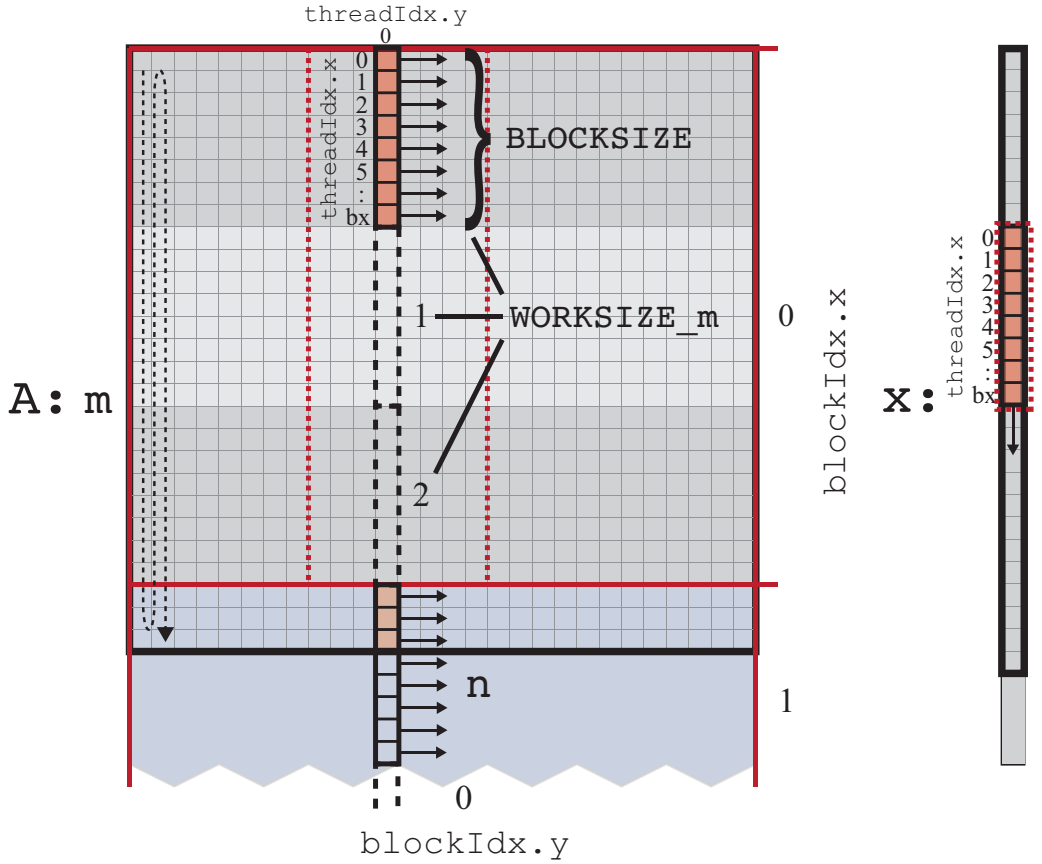
- For wide matrices; we avoid ”few threads that do all the work”.
- This requires tuning parameters: `BLOCKSIZE, WORKSIZE_n.`



Level 2 BLAS: Matrix-Vector Mult.

Several rows per thread

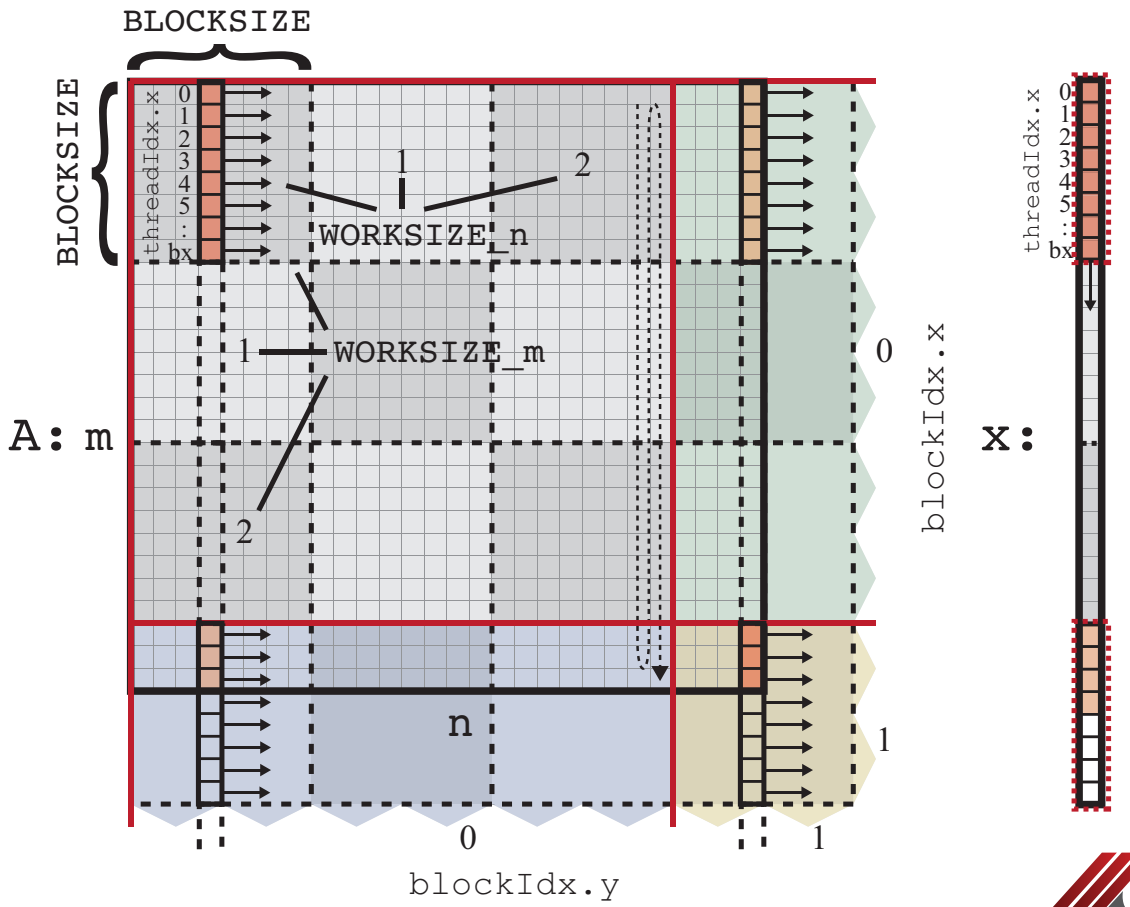
- For tall matrices; we avoid ”many threads that do little work each”.
- This requires a new parameter: `WORKSIZE_m`.



Level 2 BLAS: Matrix-Vector Mult.

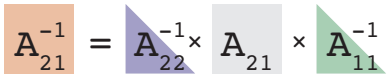
Generic kernel

- This requires parameters: `BLOCKSIZE, WORKSIZE_m, WORKSIZE_n.`

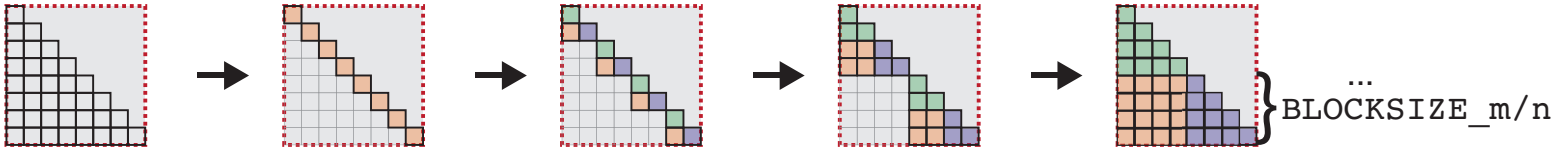


DTU Informatics

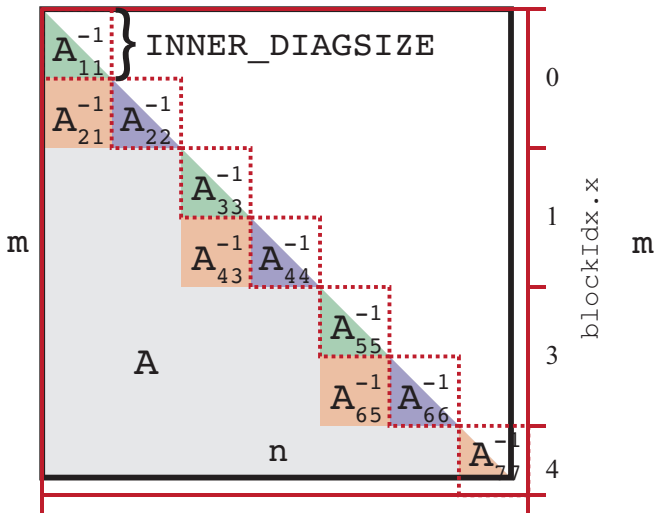
Level 2 BLAS: Triangular Solve



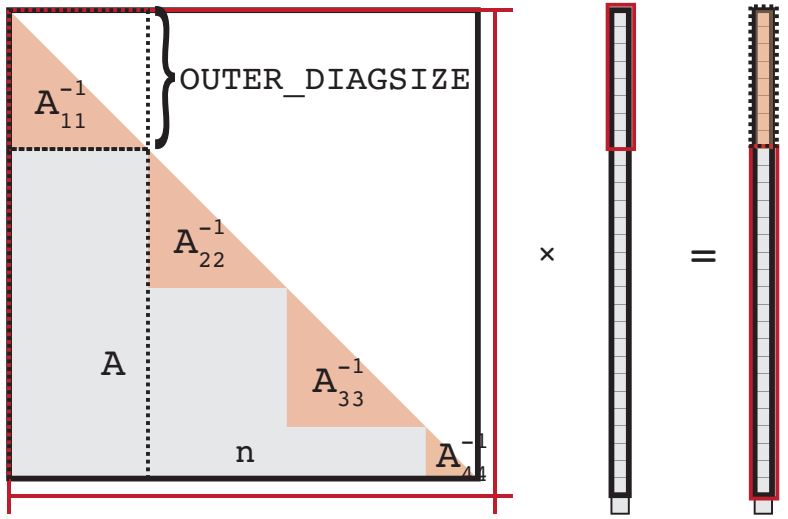
A. Triangular Matrix Inverse of Diagonal Blocks in Shared Memory



B. Expanding Triangular Matrix Inverses



C. Blocked Forward Substitution



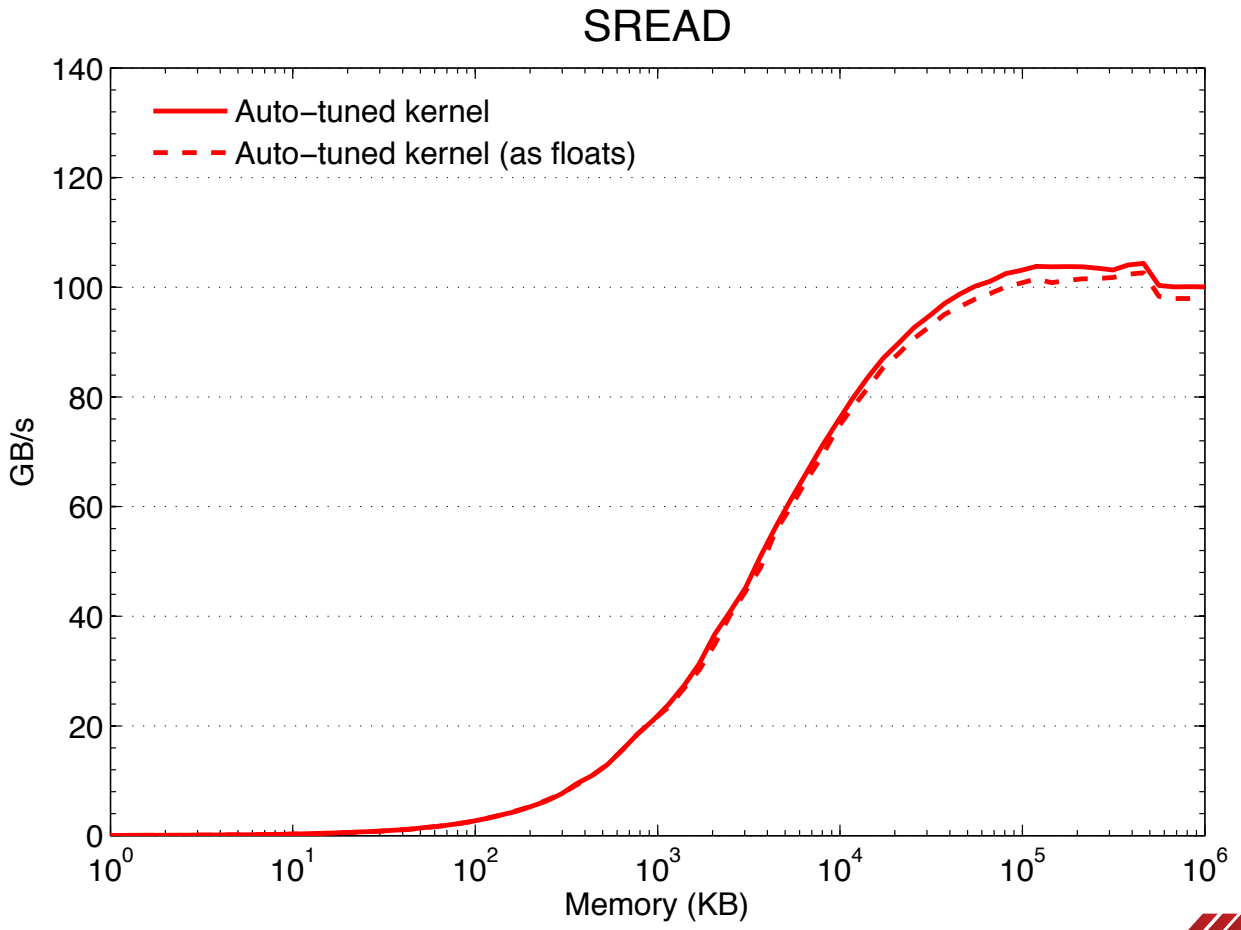
Tuning parameters: `BLOCKSIZE_m/n, INNER + OUTER_DIAGSIZE`

Performance Prediction

Performance Prediction

Tesla C2050: Theoretical bandwidth 144 GB/s

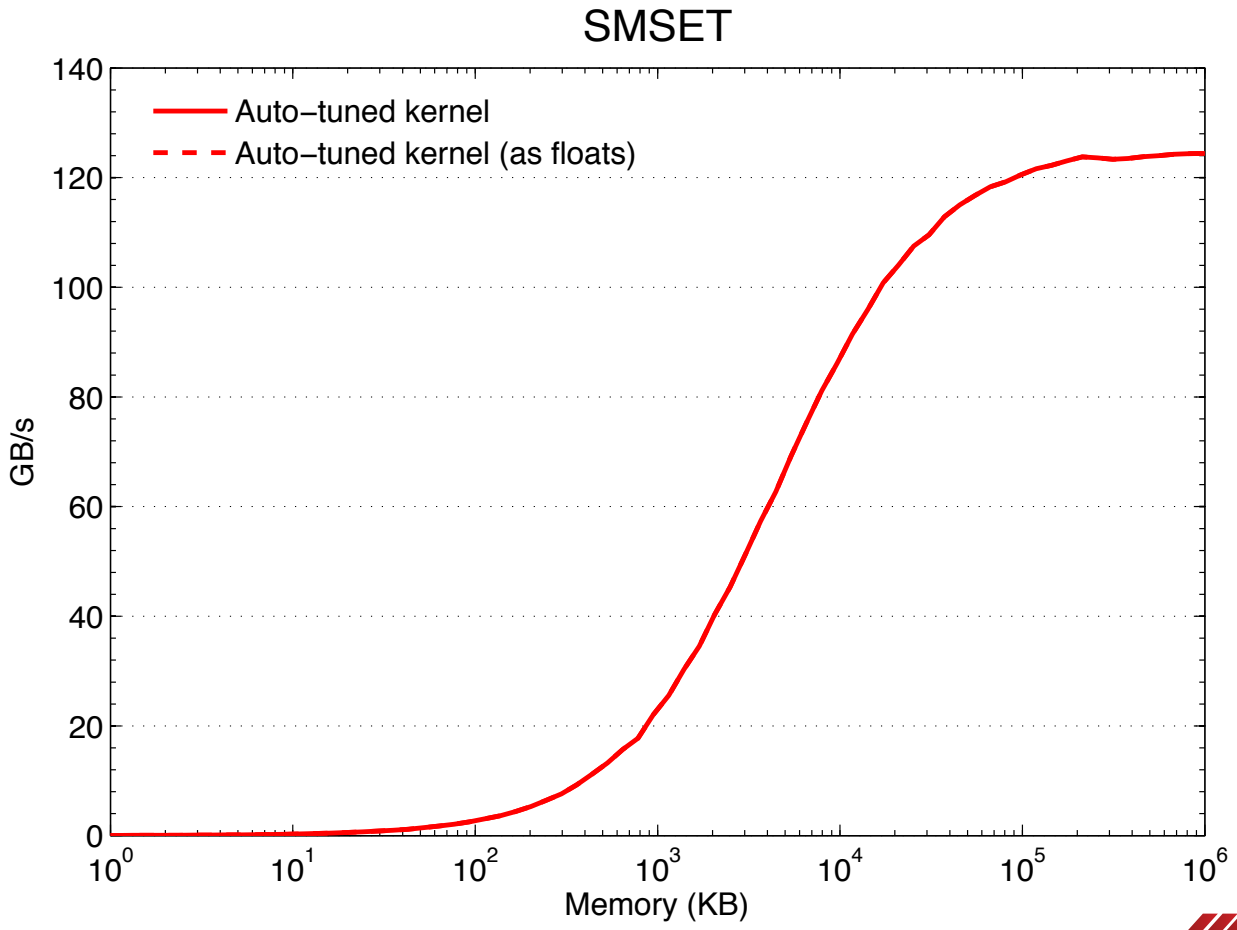
What is the effective bandwidth?



Performance Prediction

Tesla C2050: Theoretical bandwidth 144 GB/s

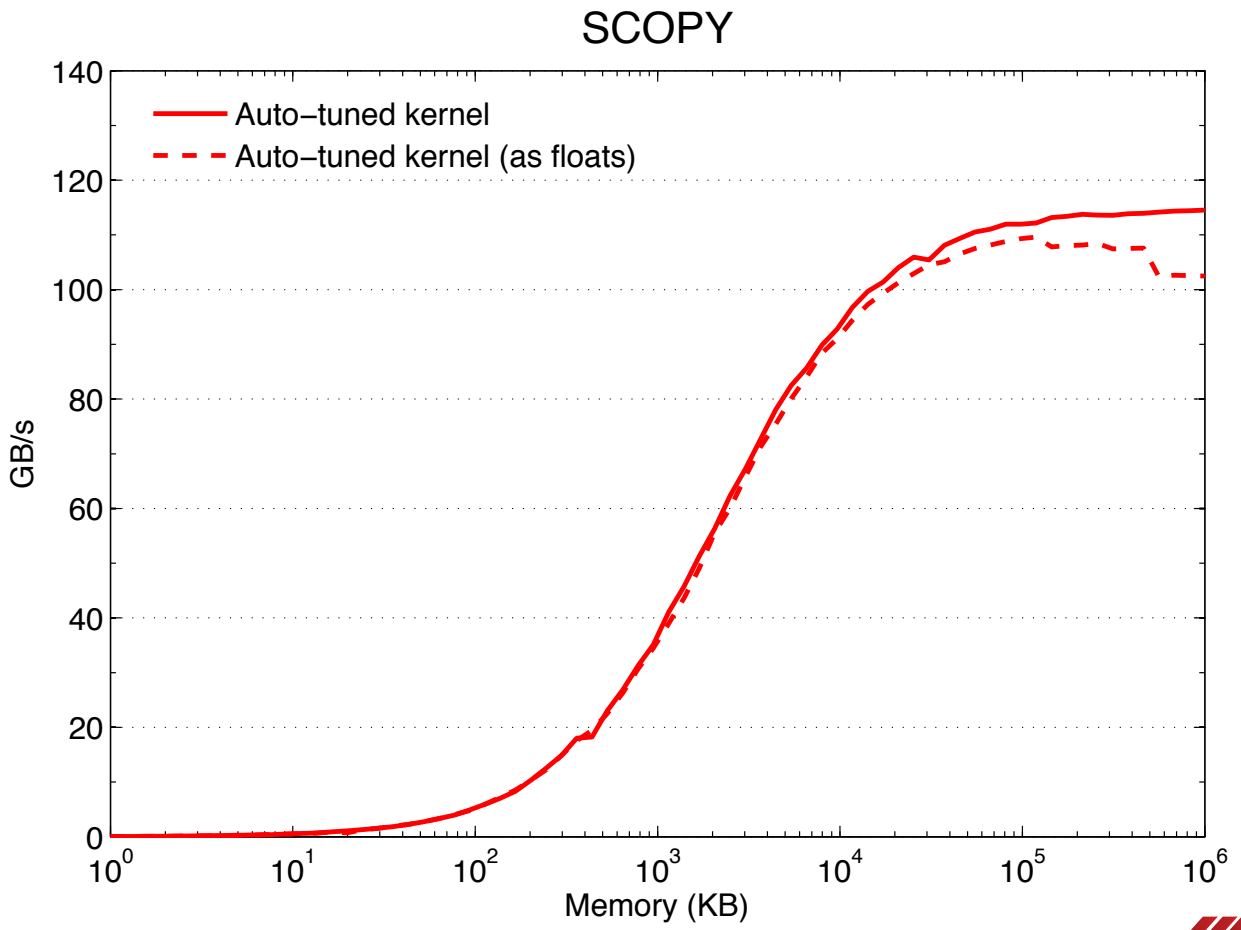
What is the effective bandwidth?



Performance Prediction

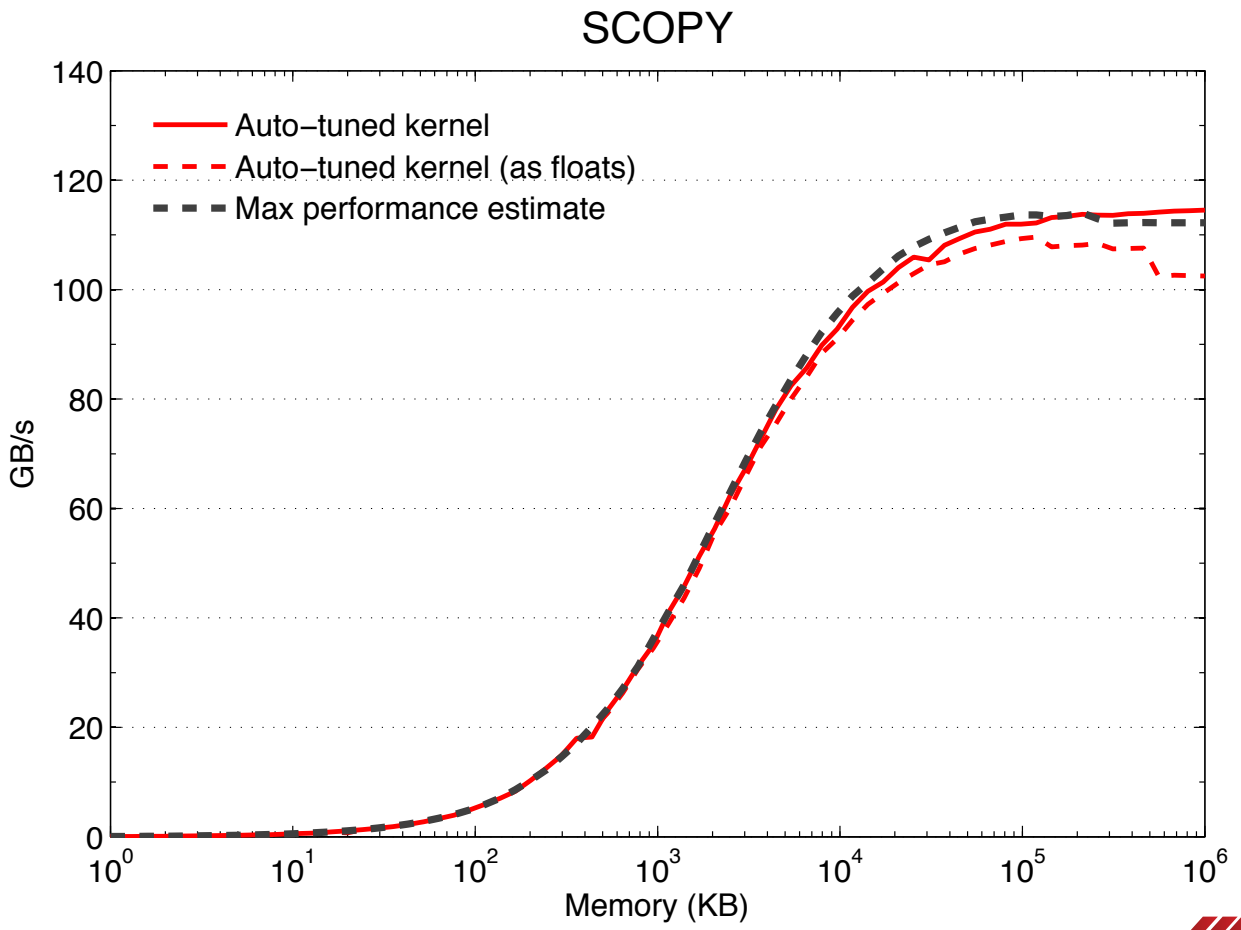
Tesla C2050: Theoretical bandwidth 144 GB/s

What is the effective bandwidth?



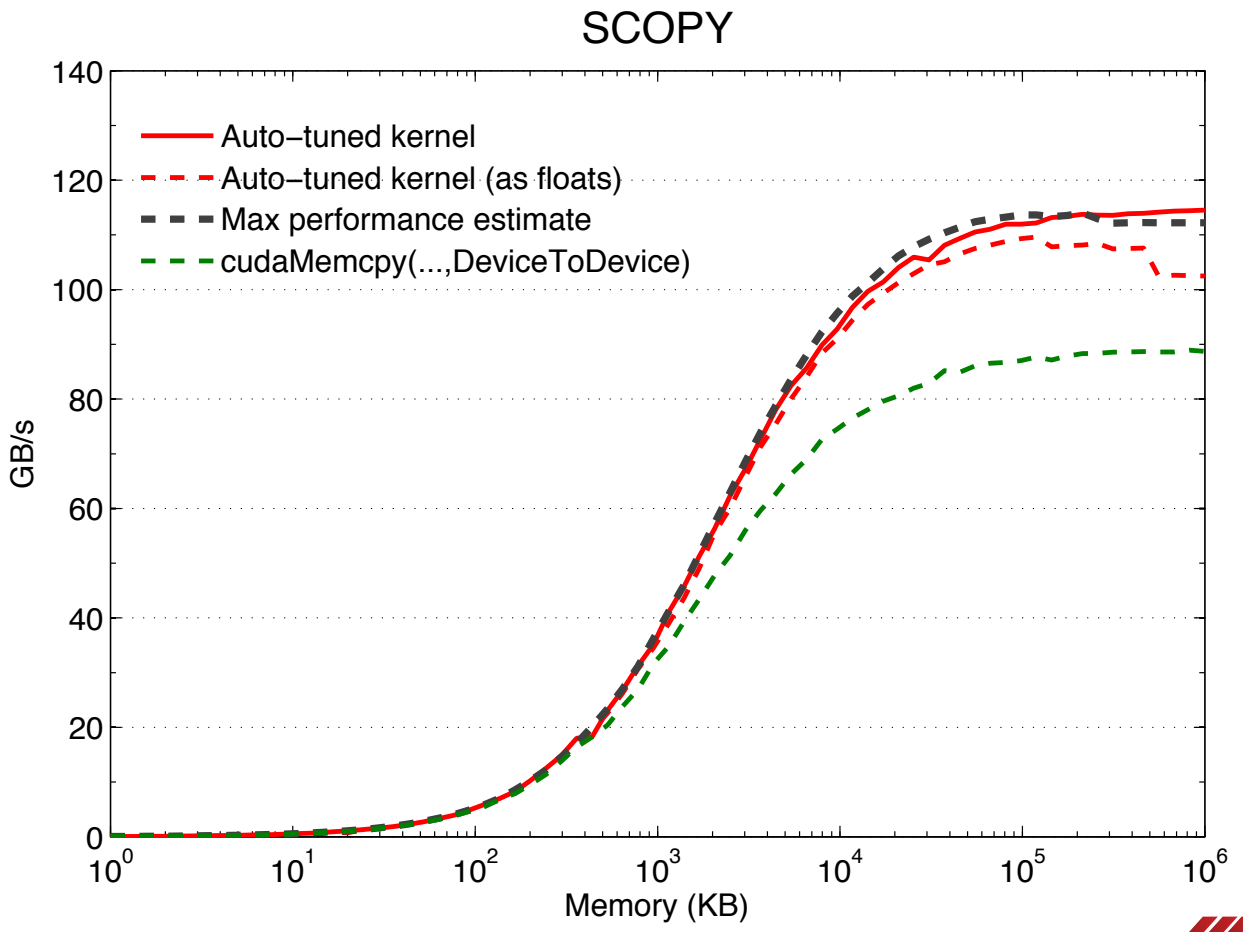
Performance Prediction

SCOPY bandwidth = (1xSREAD + 1xSMSET)/2



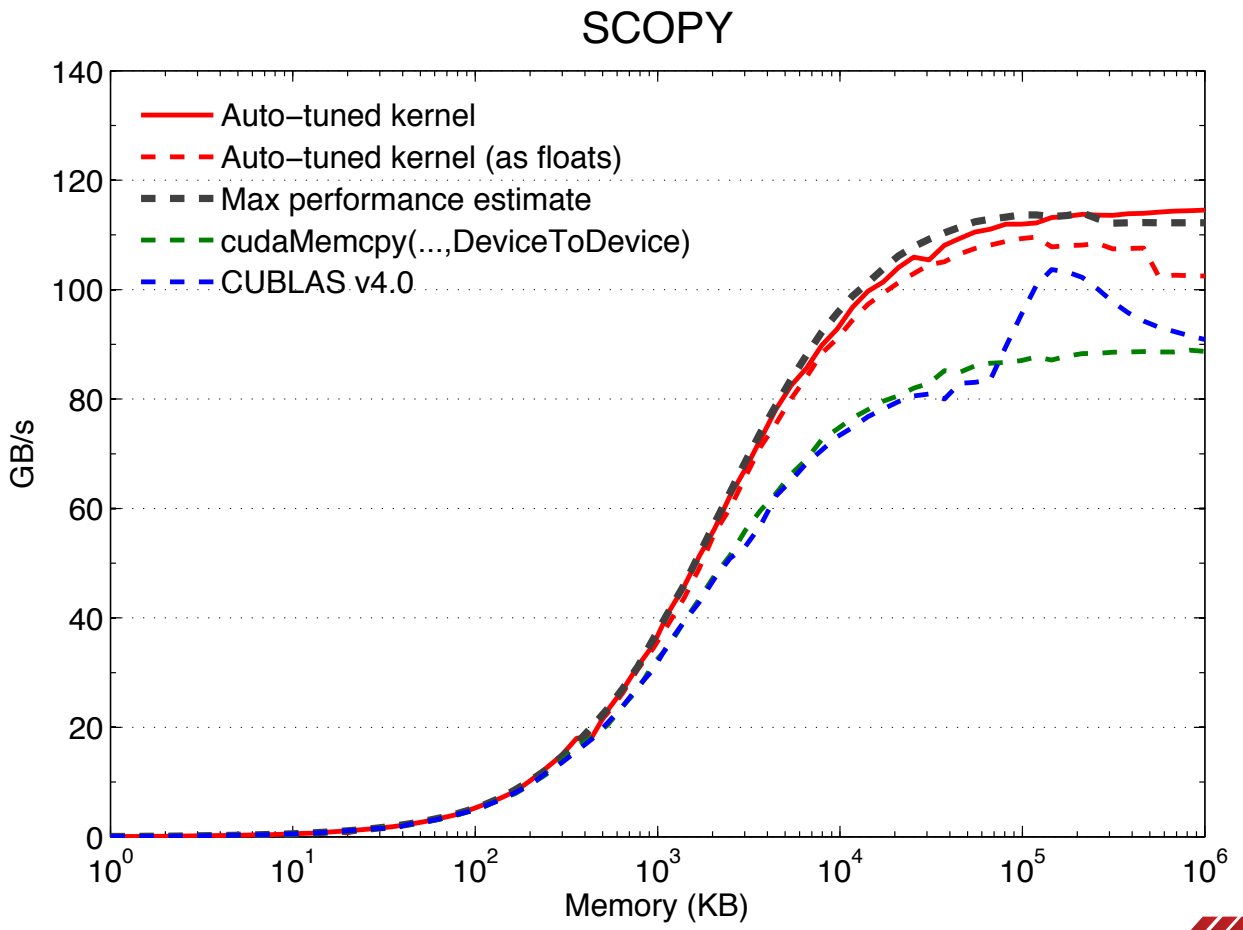
Performance Prediction

SCOPY bandwidth = (1xSREAD + 1xSMSET)/2



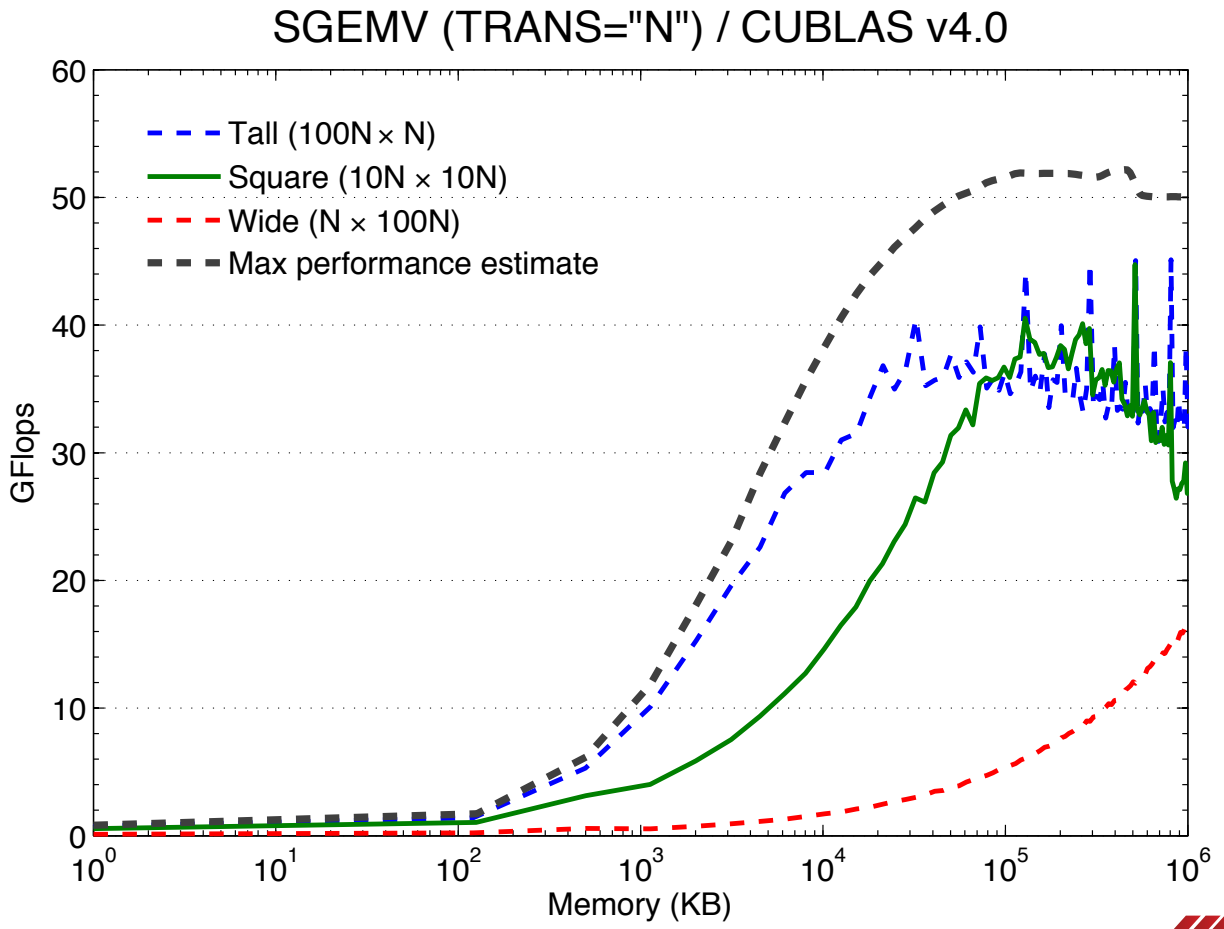
Performance Prediction

SCOPY bandwidth = (1xSREAD + 1xSMSET)/2



Performance Prediction

SGEMV = ((MxN+N)xSREAD + MxSMSET)/(MxN+N+M)



Auto-Tuning Results

Auto-tuning

- Exhaustive search of pruned parameter space (Fermi):

$$\texttt{BLOCKSIZE} \in \{32, 64, 96, 128, 160, 192, 224, 256\}$$

$$\texttt{UNROLL\_LEVEL} \in \{1, 2, 3, 4, 5, 6, 7, 8\}$$

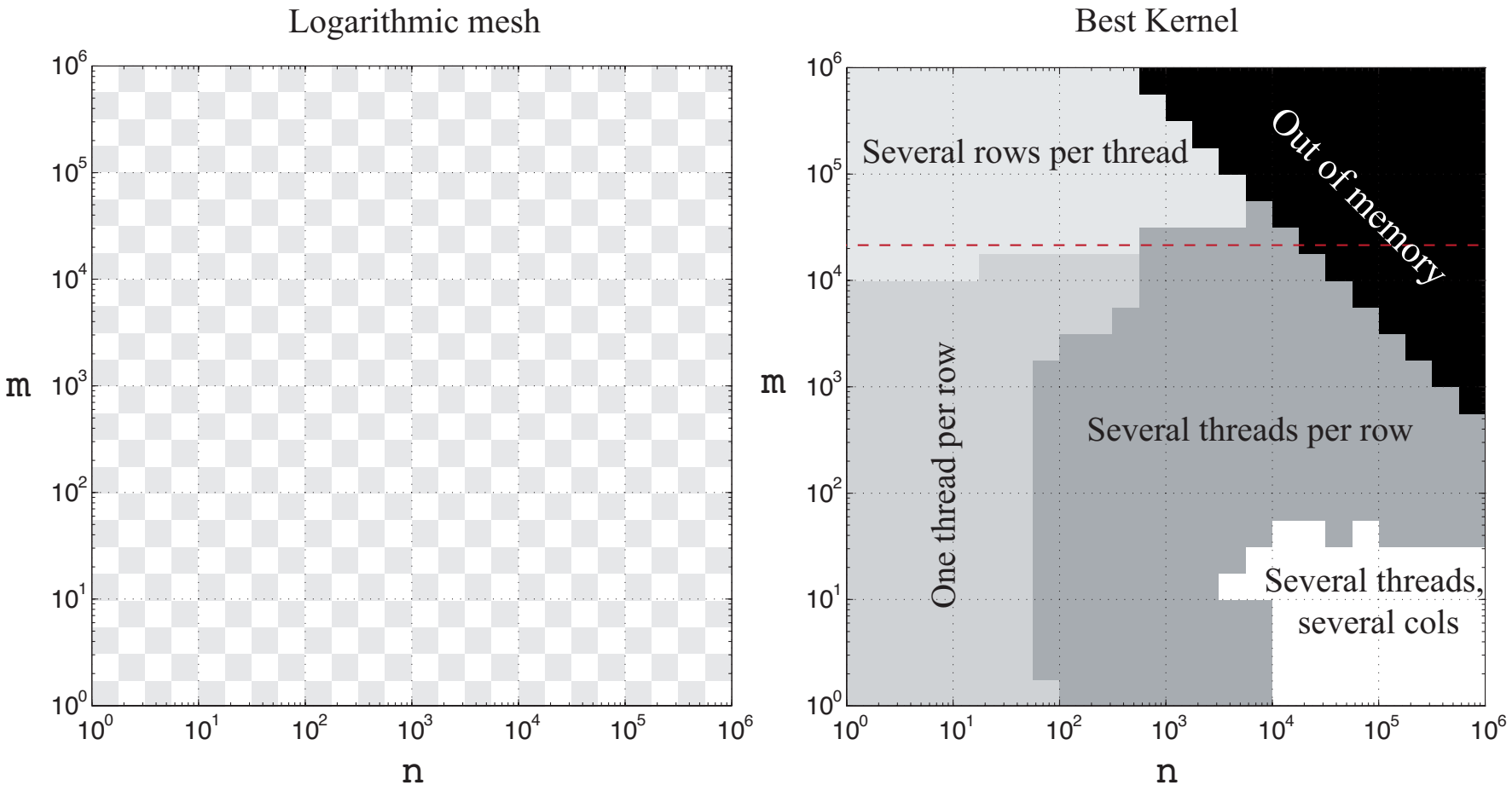
$$\texttt{WORKSIZE\_n/m} \in \{1, 2, 3, 4, 5, 6, 7, 8\} \times \texttt{BLOCKSIZE}$$

$$\texttt{INNER/OUTER\_DIAGSIZE} \in \{1, 2, 4, 8, 16\} \times \texttt{BLOCKSIZE}$$

- Up to 8^3 = 512 kernel candidates per tuning run

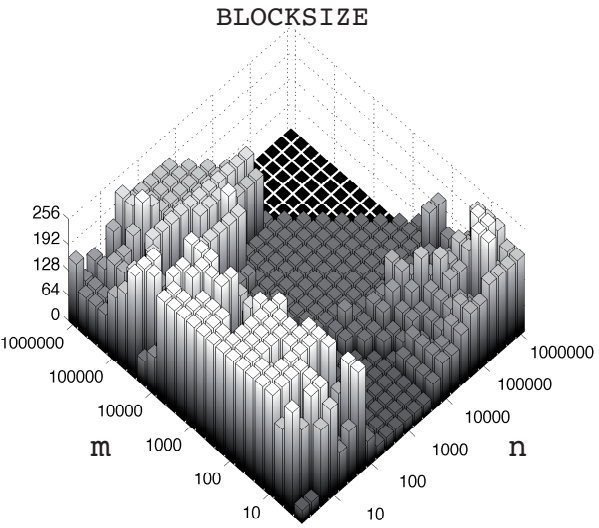
Auto-tuning

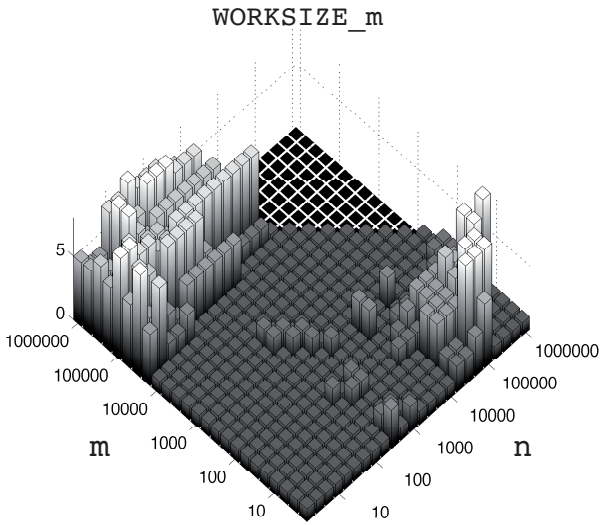
- Tuning mesh in 2D and best kernel selection for SGEMV

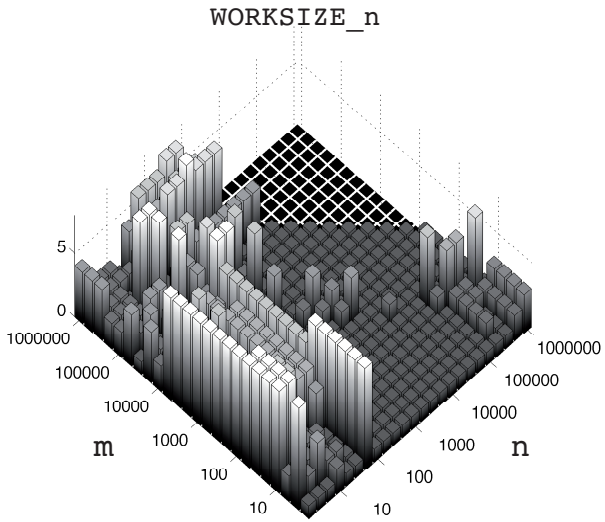


Auto-tuning

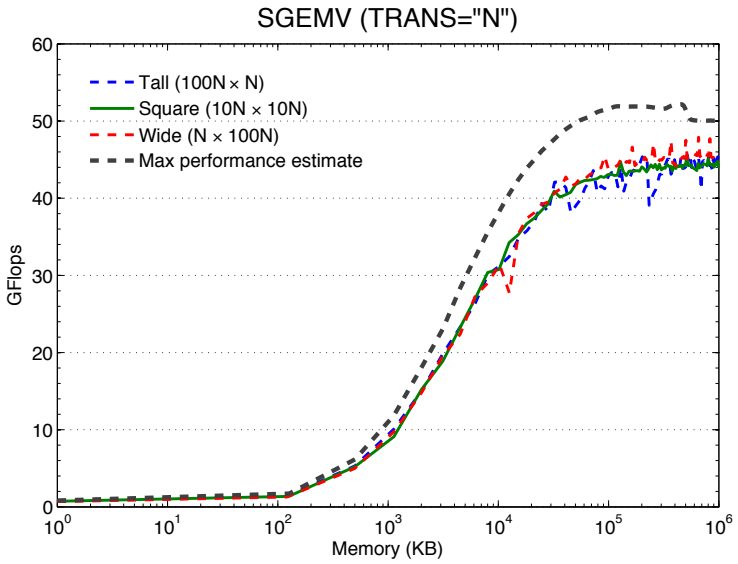
- Best tuning parameters for SGEMV

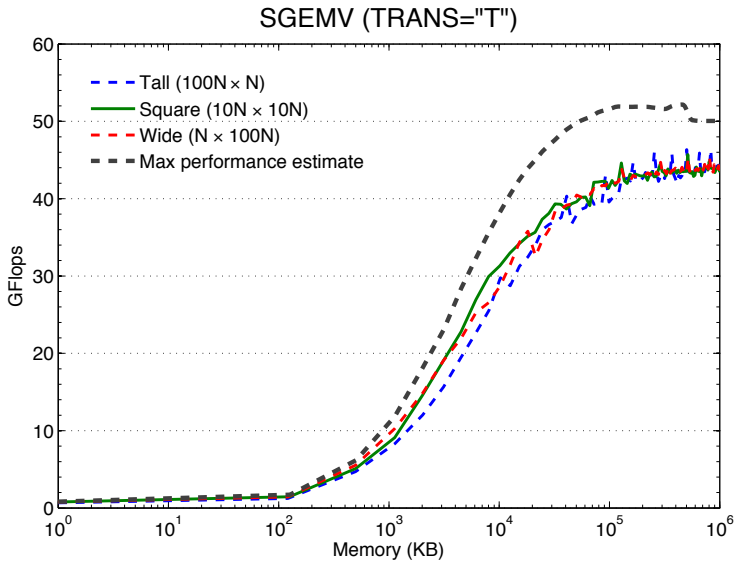


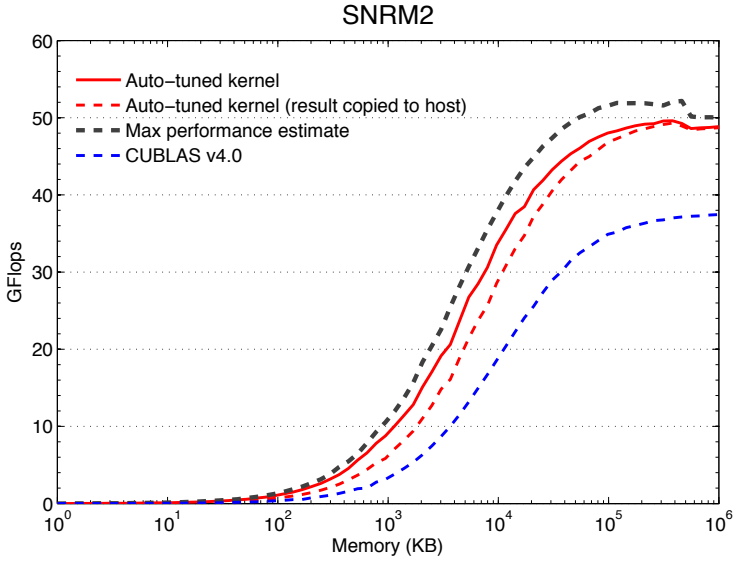


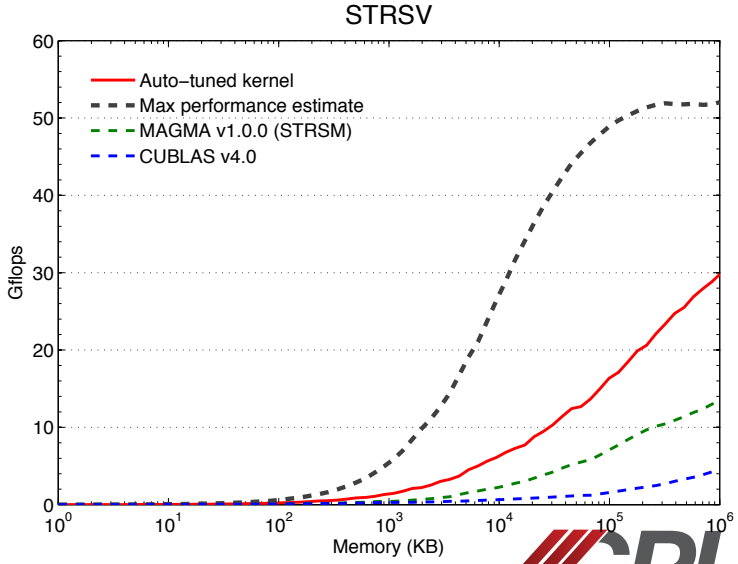


Results


SGEMV (TRANS="N")
Tall (100N × N)
Square (10N × 10N)
Wide (N × 100N)
Max performance estimate
GFlops
Memory (KB)


SGEMV (TRANS="T")
Tall (100N × N)
Square (10N × 10N)
Wide (N × 100N)
Max performance estimate
GFlops
Memory (KB)


SNRM2
Auto−tuned kernel
Auto−tuned kernel (result copied to host)
Max performance estimate
CUBLAS v4.0
GFlops
Memory (KB)


STRSV
Auto−tuned kernel
Max performance estimate
MAGMA v1.0.0 (STRSM)
CUBLAS v4.0
Gflops
Memory (KB)

Summary

- BLAS level 1 and 2 kernels are completely memory bound.
- GPU's effective bandwidth sets the max performance.
- Simple auto-tuning can facilitate high-performance Vector and Matrix-Vector kernels for all input sizes and shapes.

- GLAS for single precision is available for download at `gpulab.imm.dtu.dk` (Open Source MIT License):
  - `glas_v0.2_C2050_cuda_4.0_linux.tar.gz`
  - `glas_v0.2_GTX590_cuda_4.0_linux.tar.gz`

Autotuning: Scam or artistic instrument?